Adaptive Asynchronous Clustering Algorithms for Wireless Mesh Networks

Cheng Qiao, Kenneth N. Brown, Fan Zhang, *Member, IEEE* and Zhihong Tian, *Senior Member, IEEE*

Abstract—It is a challenge to generate an accurate machine learning model in a distributed network due to the increased concern in data privacy and high cost in gathering all raw data. This paper presents an adaptive asynchronous distributed clustering algorithm and two centralised methods for agents in wireless network to learn the global models, while the privacy is protected. Moreover, the communication cost and clustering quality can be adaptively balanced. The proposed clustering algorithms do not require the number of clusters to be pre-defined, and we propose a bounding boxes based method to fully utilize the shape information of clusters to improve the accuracy of the global model. Furthermore, we consider different knowledge levels of agents and different requirements about the global model. In experiments on randomly generated network topologies, we demonstrate that methods which do all the iterations of clustering in each cycle, and which exchange descriptions of cluster shape and density instead of just centroids and data counts, achieve higher accuracy, in significantly shorter elapsed time.

Index Terms—Distributed algorithm, Asynchronous, Clustering algorithm, Wireless Mesh Network

1 Introduction

A massive amount of data, generated by devices such as smart phones and sensors, has facilitated the machine learning (ML) methods. They are widely explored in the applications of Internet of Things (IoT) [1–4]. For IoT system, Wireless Mesh Networks (WMN) are becoming increasingly important in many applications [5–7]. According to the research report from IDC, the global datasphere could grow up to 175 zettabytes (ZB) by 2025, and IoT devices are expected to create over 90 ZB of data in 2025 [8]. However, data are usually isolated and it is a challenge to centralise all data to a central server due to the privacy issue and the high cost [9].

Distributed learning is proposed to address this problem [10, 11]. One distributed learning technique, named Federated Learning (FL), is widely applied in various applications for its efficiency and privacy benefits, where agents in the network learn or train their models locally and only periodically exchange models with a central server [12]. However, this centralised method can not be implemented directly due to the limited wireless resources (i.e., transmission range and non-rechargeable battery). Therefore, it is necessary to consider a new framework in wireless network, where agents in the network exchange model with its neighbours (hereinafter called *distributed methods*) or a designated agent (hereinafter called *centralised methods*) instead. Our intuition is that this in-network learning, by reducing reliance on the server, can reduce high communication cost, respect the privacy and potentially improve robustness (e.g., agent failures).

In the context of wireless network, the shortage of labeled data severely hinders the applicability of supervised learning [13]. In this paper, we concentrate on the distributed clustering problem [14, 15]. Many works have explored techniques for distributed clustering (i.e., federated clustering, parallel and distributed clustering), but most do not take into account the challenges of wireless networks (for instance, systems heterogeneity, dynamic networks and limited resources), clustering algorithms (i.e, how to choose the number of clusters $\boldsymbol{k}$) and actual demand from end-users. While many methods aim to address these concerns individually, in this paper, we suggest that those deficiencies can be addressed by a unified framework, adaptive asynchronous clustering algorithm (AAC).

Different knowledge levels of the network (e.g., the size of network) plays a crucial part in the performance of distributed and centralised methods. As the agents may leave and join in a network dynamically, it is hard to know the actual size of the network in some applications (e.g., sensors deployed in a grassland for tracking the populations of animals). Thus, we consider two different scenarios for the initial state of network: 1) Each agent knows the size of network, i.e., the number of agents in the network, or 2) Each agent does not know the size of the network and is only aware of its immediate neighbours.

Personalization from end-users is another important concern for the algorithm. It poses a crucial role in the overall performance of network. In particular, it may be unfair to compute a single global model for all agents when local datasets are sufficiently private and the data distribution is non-IID [16]. Thus, we consider two assumptions about the final clustering model: 1) all agents may finish with slight different but similar models, or 2) all agents must finish with identical models. Figure 1 shows different scenarios considered in this paper.

Example 1: Suppose some sensors are deployed in a forest to monitor the temperature. We could centralise all collected readings to a designated sensor and compute the average temperature (for the case of centralised methods C_B and C_R). Alternatively, sensors could cluster its local readings first and share the clusters with its neighbours (for the case of distributed methods). Given different settings about the network and requirements from end-users, the procedure for distributed methods is different. For

- *Corresponding author: Zhihong Tian (tianzhihong@gzhu.edu.cn).*
- *Cheng Qiao, Fan Zhang and Zhihong Tian are with Cyberspace Institute of Advanced Technology, Guangzhou University, China. Email: qiao.cheng@insight-centre.org, zhangf@gzhu.edu.cn and tianzhihong@gzhu.edu.cn*
- *Kenneth N.Brown is with Insight Centre for Data Analytics, Department of Computer Science, University College Cork, Ireland. Email: k.brown@cs.ucc.ie*

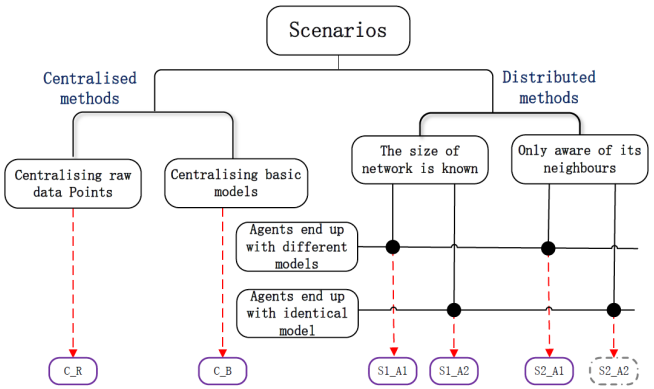


Fig. 1: Different scenarios considered in this paper

instance, how to learn the average temperature in this area when there are N agents in the network (for the case of $S1\_A1$), how to learn the average temperature computed by the first agent in this area when there are N agents in this network (for the case of $S1\_A2$) and how to learn the average temperature when the size of network is not known (for the case of $S2\_A1$).

Intuitively, the data are partitioned in an non-IID fashion in heterogeneous networks, since the distribution of data on each agent may differ. So we expect that the number of clusters for some agents (or all agents cross the network) is different. However, all algorithms that we discussed, including existing state-of-the-art [17–20], assumed that the number of clusters $\boldsymbol{k}$ is known by all agents in advance. Therefore, an adaptive asynchronous clustering algorithm is required, which can derive the best number of clusters as part of the algorithm. In this paper, we take the system heterogeneity into account and assume that each agent estimates the initial $\boldsymbol{k}$ value based on its local raw data independently, and adjusts this value constantly.

The proposed algorithm AAC allows each agent to learn a global picture and take appropriate actions. Each agent acts as its own decision maker, but must communicate with neighbouring agents to learn wider network patterns [21]. Each agent represents its raw data points as a cluster model (the number of clusters is not necessarily consistent), and shares this model with its neighbours. Agents must combine the shared models into a single description, and eventually, all agents agree with the description (similar or identical). Cluster centroids and counts are used in existing studies to summarise the description of data distributions of cluster [17, 19, 20]. Transmitting clusters centroids and counts is not enough, as it loses important information about the distribution of data points around the centroids.

At some stages in the overall algorithm, an agent will need to combine cluster models from two or more agents. The weighted average scheme is widely used for this aggregation procedure [17, 19, 20]. Note that the underlying data distribution for this approach is assumed to be uniform. Furthermore, this method cannot be used when the number of clusters for each agent in the network is not the same. We consider a regeneration based method to update centroids and counts. Agents generate new sets of data points by sampling from the distributions or descriptions, and then re-running local clustering.

We develop an asynchronous mechanism for sharing information, to avoid synchronisation delays. Different approaches to in-network clustering, and trade-offs between sharing local clusters and incremental clustering as information flows through the network are considered. We design and implement different methods for describing the shapes of locally identified clusters, to improve the global clustering performance. The proposed methods are evaluated empirically, using an asynchronous message delay simulator, and the evaluations show that the new approaches can improve clustering accuracy compared to existing methods (relative to a centralised solution) by up to 10%, while decreasing the cost of message transferring by 15% and dropping convergence time by up to 85%, all without transmitting any raw data.

This paper is an extension of our previous conference paper [22]. The new contributions in this paper are summarized as follows.

1) We design, implement and evaluate adaptive asynchronous distributed clustering algorithms that allow each agent to learn the global view, which relaxes the assumption that the number of clusters is fixed in advance. Furthermore, we consider two different requirements about the final cluster models from end-users: identical or similar.
2) We also propose centralised algorithms to learn the global cluster model. Agents could centralise all raw data points or basic models to a designated internal agent.
3) We consider three different underlying synthetic datasets (from heavily overlapped to well separated) and three real-world datasets, to evaluate the performance of the proposed algorithm.
4) We introduce a new method to measure the accuracy. This new measurement will not penalise the algorithm if the learned model does not fit its local raw data well.

In the remainder of the paper, we survey related works in next section, then define our methods, followed by the experimental set-up, and finally present and discuss the results of the experiments.

2 RELATED WORKS

FL is gaining more and more attention for the benefits of privacy and security, and zhang et al. [23] presented a comprehensive survey of various applications of FL. Personalization in FL is crucial for some practical applications, and it is explored in multi-task [18] and meta-learning [24]. Mansour et al. [16] proposed three different approaches to compute the personalized models in the context of FL: 1) central agent trains the same model for a group of clients (cluster), 2) agent adds a hyper-parameter λ to balance the effect caused by different size of global data and personal data, 3) agent adds an interpolated weight λ_k to merge the global model and local model.

For distributed data mining, in which more general patterns must be inferred, clustering algorithms are mainly used in two directions: design the routing protocol [25, 26] and compute the local model of individual agent [17, 18, 27]. For planning the routing, the basic idea is clustering agents as groups to reduce the network overhead. Those groups are managed by cluster heads, which gather data from nodes inside the group, aggregate the data, and then transmit summaries as needed to the central base station [25]. However, deciding which agents should be cluster heads is a challenging task [28]. Vural et al. proposed an asynchronous clustering method for multi-hop Wireless Sensor Network, where the cluster head is elected based on residual agent energy levels and traffic loads [29]. It only requires synchronisation in local area and can be extended to different types of network topology, but a hierarchical network is assumed.

Clustering is also used in computing the local representation of agent for large-scale peer-to-peer networks, which substantially reduce the energy consumption of message transmission. Distributed K-means was proposed in [17, 18], where the K-means algorithm is distributed by inserting an exchange of messages between neighbouring agents after each internal iteration of the basic K-means loop (the selection of centroids, and the assignment of individual data points to each centroid) on the agent's local data. In this scheme, no raw data is exchanged, thus satisfying the basic privacy requirements, and the centroid summaries reduces the amount of data being transmitted. However, the algorithm requires synchronisation, which in wireless networks may introduce significant delay, and may fail to converge if one agent has initial data whose distribution is significantly different from the average. The exchange of just centroids and counts also discards information about the shape of the cluster. When clusters have significantly different shapes, this may again cause problems for convergence. Another method is coreset-based method [30, 31]. The main idea is constructing coresets for each partition of the whole data set and representing the whole data set as coresets.

Gossip-based aggregation methods for distributed K-means are proposed by Fatta et al. [19] and Bénézit et al [20]. Each agent selects a small subset of its neighbours to communicate with at each round, and includes a damping factor in the weighted average to avoid oscillations. Although global synchronisation is avoided, local synchronisation is still required. In practice, the nature of the gossip algorithm means that many rounds are required before convergence.

The most related work in the literature is [32], in which the authors propose a decentralized and adaptive K-means clustering for Non-IID data. To avoid the problem of data over-represented, the hyperLogLog counters are employed to approximate the total number of distinct data points involved in the weighting process. The number of clusters is re-estimated constantly by further clustering the centroids. However, representing the clusters as centroids only will lose important information about how data points spread inside the cluster. Thus the re-estimated number of clusters relies on the centroids will be biased. In addition, only clustering accuracy is measured. Note that the low accuracy of approximate algorithm hyperLogLog used to track the unique data points will undoubtedly degrade the model [33].

To address the cluster shape issue, for tree-based network topologies, Bendechache et al. [27] represent each cluster by its boundary points, and then exchange the boundary and the number of internal points. Messages flow up the tree to the root. When an agent receives multiple descriptions, it computes a new description by merging any overlapping clusters into a single cluster, and computing the new boundary. The algorithm is initialised with a much higher k value than is expected, to allow this cluster merging to reduce the number of clusters. This approach solves the issue of arbitrary shapes, but introduces failure points at each aggregation, and requires a secondary communication from root to leaves to disseminate the final clusters. It also exposes some of the raw data to neighbouring agents, in order to specify the cluster boundary.

The assumption that the number of clusters is known by all agents in advance, is adopted by most existing state-of-the-art work [17–20]. However, in most practical applications, there is no prior information on how many clusters there should be. Therefore, an adaptive asynchronous clustering algorithm is required, which can derive the best number of clusters as part of the algorithm. We could simply rerun the algorithm for each possible $\boldsymbol{k}$ value, but this would be expensive in both elapsed time and in communication cost. We note that one recent paper on distributed clustering, [27], does derive the appropriate $\boldsymbol{k}$ value automatically, by starting with large $\boldsymbol{k}$ values, and then combining multiple clusters together as information flows up a tree in the network. We focus here on algorithms for a flat network. For comparison, we also consider methods which send all raw data or basic models to a single identified agent.

3 AAC Framework Description

3.1 Overall Structure

In this section, we show the overall framework, where distributed method and centralised method are mainly described. We assume a flat ad hoc communication network with no specific topology, and no distinguished agents. Agents observe some triggers to start clustering - the trigger could be synchronized across the whole network, or could be local to one or more agents. If an agent receives a request to exchange its cluster models, it will interpret this as the trigger to start clustering.

3.1.1 Distributed methods

For distributed methods, we describe the overall structure of adaptive asynchronous clustering algorithm (AAC) for all agents to learn wider network patterns (see algorithm 1). As we mentioned before, we aim to propose a unified framework and address the challenges of wireless network, clustering algorithms and actual demand from end-users.

The knowledge level of network size will change if the network is dynamic. Thus, we consider two levels of agents' knowledge: the size of the network is known in advance or agents only know their immediate neighbours. Note that the clustering algorithm itself plays an important role in the overall performance of network and it is indispensable to decide the number of clusters in advance. We consider two different assumptions about the number of clusters: the number of clusters $\boldsymbol{k}$ is pre-defined and agents estimate the $\boldsymbol{k}$ locally and independently.

Algorithm 1: Asynchronous distributed clustering algorithm for agent N_i

```
1 Input: knowledge level S_l, number of clusters k, local
         dataset X_i
2 Initialization: generate initial cluster model;
3 Exchange cluster descriptions with neighbours;
4 Wait until messages received or time exceeded ;
5 while time not exceeded and messages received do
6     Incorporate received neighbours' cluster information;
7     Run local clustering to completion;
8     Transmit new summaries to neighbours;
9     Wait until message received or time exceeded;
```

The basic steps are as follows: an agent A_i performs a local clustering algorithm based on the knowledge level S_l and the number of clusters $\boldsymbol{k}$ (line 2). A_i summarizes the local clustering result then transmits it to A_i's neighbours (line 3). After receiving cluster summaries from all neighbours, A_i executes the local clustering algorithm again with the new information (line 7). Once A_i detects no significant changes (measured by a threshold) comparing to the clustering in the previous iteration and there are no incoming messages from other agents, AAC terminates (line 9). We explore these steps in more detail in Section 3.2.

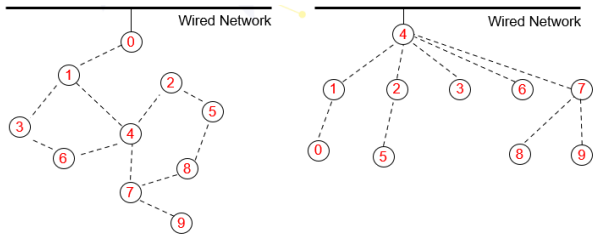
3.1.2 Centralised methods

The centralised method is also a potential method to learn the network wider pattern. Agents in this network forward their observations or summaries to a designated agent, and the global model is computed. Then a secondary communication from this designated agent to all other agents is required to disseminate the final model. Based on this procedure, we assume that all agents know the size of network ($S1$) and an identical global model ($A2$) is required by the centralised methods.

We proposed two new centralising methods to learn the global model. These two centralising methods are based on a tree based topology, which differs from the flat topology used in distributed methods. The central agent is picked by centrality. Finding out which is the most central agent is crucial, because it could help to disseminate information in the network faster and protect network from breaking. We decide the central agent by closeness centrality because it is very intuitive and suitable to characterise a process in which the information travels through the shortest distances. Formally, the closeness centrality of i is defined as:

$$C_i = \frac{N}{\sum_{j=1, j\neq i}^{N} d_{ij}} \tag{1}$$

where d_{ij} is the length of the shortest path between i and j in a N-agent network. Figure 2 shows an example how to construct an tree structure based on the agent centrality.



(a) Initial state of the network root (b) Tree structure of network root

Fig. 2: Tree structure based on centrality

For this tree based structure, we consider the following two approaches:

1) Centralise raw data points to a designated central agent. For individual agents, it requires multiple packets to send out all its raw data points. We consider two different assumptions: in the first, there is no inter-packet delay, while in the second there is a uniform inter-packet delay of 0.1s.
2) Each agent produces its own basic model as before, and then transmits that model to the designated central agent.

To do this, we have to define the routing table for agents. We assume that this routing table is available for each sensor in advance and the computation cost to compute the routing path is ignored here. A shortest-path tree is constructed with the central agent as the root. For different types of agents, the sharing strategy is listed as follows:

1) Root agent: wait until all information is received from its children, and then compute the final model.
2) Leaf agent: send all messages to its parent agent (raw data points or basic models).
3) Other agents: send all its local data to its parent agent and whenever messages are received from child agents, relay the message on to the parent.

3.2 Descriptions of Local Clustering Algorithm and Cluster Integration

In the last section, we described the general framework for distributed methods and centralized methods. These two methods do not require any prior information about the underlying distributions. In this section, we describe the procedure and related technique for each step outlined in algorithm 1.

3.2.1 Estimate the number of clusters k

The initial step is to choose an appropriate $\boldsymbol{k}$ based on its local data. There is no clear way to choose the appropriate number of clusters, since the shape and scale of the distribution of points in a dataset and the desired clustering resolution vary with applications. Note that increasing the number of clusters $\boldsymbol{k}$ will always reduce the amount of error (for example, Mean Squared Error), but the performance, usually measured by minimal intra-cluster distance and maximal inter-cluster distance, would be degraded when $\boldsymbol{k}$ is getting large until we reach the extreme case that $\boldsymbol{k}$ equals the number of data points.

Although choosing an appropriate k for clustering algorithms is well studied, it has not yet explored in the context of distributed learning. The assumption that the number of clusters is known by all agents and fixed to be the same for all agents in advance, is adopted by most existing state-of-the-art works [18, 19, 19, 20, 27], then a weighted averaged scheme is used to update the centroids. However, in most practical applications, there is no prior information on how many clusters there should be. We propose that each agent estimates its $\boldsymbol{k}$ locally and independently, which means that some agents may have different $\boldsymbol{k}$ values.

Olatz et al. shows that Silhouette achieves the best results among the 30 cluster validity indices in most cases [34]. It measures how similar a point is to its own cluster (cohesion) compared to other clusters (separation). The cohesion is measured based on the distance between all the points in the same cluster and the separation is based on the nearest neighbour distance. The silhouette value ranges from -1 to $+1$, where a high value indicates that the object is well matched to its own cluster and poorly matched to the nearest clusters. In this paper, we use the Silhouette method to pick the number of clusters $\boldsymbol{k}$.

We proposed that a regeneration based method (see Section 3.2.4) to integrate message from neighbours, which does require the number of clusters $\boldsymbol{k}$ for all agents is consistent. Agents could estimate and recover neighbours' datasets by exploring the summary descriptions. The recovered dataset is then combined with its local data. A new $\boldsymbol{k}$ is estimated based on this combined dataset. Since this combined dataset will update constantly, the $\boldsymbol{k}$ will update correspondingly. Algorithm 2 shows that the silhouette method used to pick the $\boldsymbol{k}$. Given a combined dataset X_c with a range of number of clusters, the $\boldsymbol{k}$ value that maximises the average silhouette will be the provisional best number of cluster. This silhouette method can be automated, since it is simple to determine the maximum value [35].

3.2.2 clustering and cluster summaries

After the confirmation of K value, the next step is doing clustering and describing the clusters. Transmitting cluster centroids and counts is not enough, since it loses important information about the distribution of data points around the centroids (see Example 2). Intuitively, the cluster shape descriptions could improve the performance of proposed algorithm (i.e., fewer messages

Algorithm 2: Method to pick the number of clusters

Input: Range of number of clusters: $\boldsymbol{k} = \{k_0, k_1, \cdots, k_n\}$, The combined dataset X_c ;
Output: The number of clusters that fits model best k_b;
for $i = 0; i \leq n$ **do**
    Compute clustering algorithm for different values k_i ;
    Calculate the average silhouette (S) on dataset X_c;
Plot the curve of S for all different values k_i;
Find out the location k_b with maximum S in the plot;

transmission and higher accuracy). In this paper, we propose that each cluster is described by centroids, estimation of size, and a description of distribution and shape instead, which fully describe the clusters.

K-means and GMM are two widely used clustering algorithms since they are robust, computationally fast and easy to implement. We use these two algorithms to compute the local model for agents. Although GMM describes the cluster as mean and variance, it assumes that the underlying distribution of cluster is normal (denoted as GMM). This motivates us to find a new method to describe the shape of cluster. In this paper, we use K-means to cluster the dataset and describe the cluster as nested bounding boxes (denoted as K-means). Note that an intermediate method is finding clusters by distance and describing the clusters as mean and standard deviation (denoted as SG). Figure 3 shows the three different schemes:

1) K-means. Local data are clustered by K-means. Nested bounding boxes are then used to describe the shape of clusters. Centroids, counts and bounding boxes are shared with neighbours.
2) Separate Gaussians (SG). Similarly, local data are clustered by K-means and then a separate multi-dimensional Gaussian is fitted to each cluster, where the shape of cluster is denoted by mean and variance. In this case, the Gaussians, plus a count of the data points in each cluster, are shared with neighbours.
3) Gaussian Mixture Model (GMM). A Gaussian Mixture Model is fitted to the local data by the EM algorithm [36]. The shape of each cluster is described by the means and variances of the GMM. The GMM, plus a count of the number of data points in each cluster, is shared with neighbours.

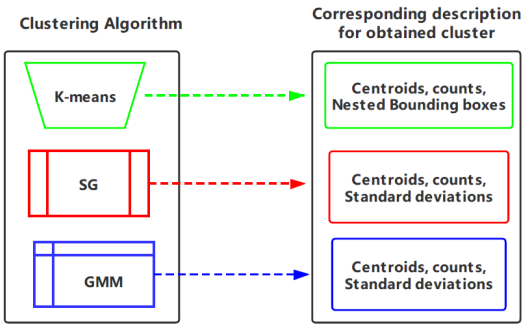


Fig. 3: Clustering algorithms and corresponding description of clusters

To generate the shape and density description for clusters obtained by K-means, we first apply Principal Component Analysis (PCA) to each cluster [37], to generate the axes of the cluster shape, centred on the centroid. For each positive and negative axis, we generate the maximal data point, and some other intermediate percentile points. We then adjust these percentile points to midway between that point and next closet point. Taken together, this produces several bounding boxes (or hyperrectangles), containing the maximal data point and other percents of the points respectively, oriented along the PCA axes, and thus approximates the shape and density of points in the cluster. Figure 4 shows an example generation of bounding boxes for a 2-dimensional dataset, where 100%, 80% and 40% are used as percentiles.

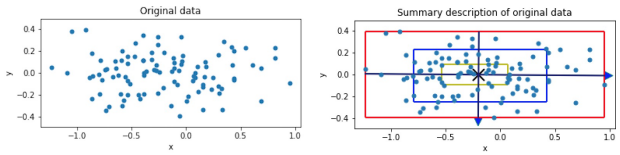


Fig. 4: Original data and its summary description

3.2.3 Data exchange

The next step is to share these obtained representation of clusters with neighbours. While an agent is doing local clustering, more messages may arrive from neighbours. We considered two approaches: transmit the result of local clustering, and then read any stored messages and repeat; read stored messages and repeat clustering, until the inbox is empty, and then transmit the clustering results. In practice, transmitting before reading new messages produced better performance, so we only report those results [1]. As well as transmitting their own summaries, we allow each agent to send on summaries received from other agents without modification if needed. Further, to each summary we attach a list of all agents whose data was used to generate the clusters. This supports the delayed clustering above, where an agent waits until it has received information from all known agents. In some cases, we also allow an agent to issue a request for reduced data summaries (summaries built from a specific subset of agents). Finally, we consider the case where messages are sent to all direct neighbours [17, 20], and where messages are sent to a randomly selected subset of neighbours [19].

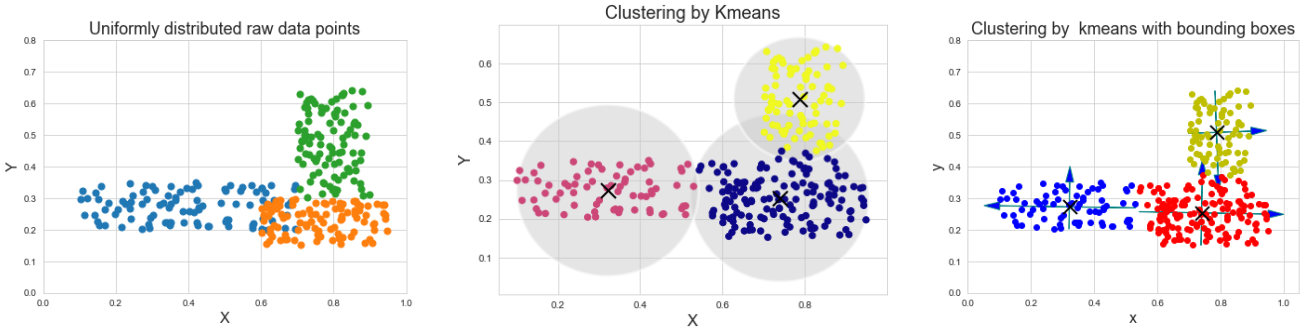
Example 3 - delayed clustering : agent A_i waits for message from agent A_k who are two hops away, then this message could be forwarded by agent A_j who connects these two agents, since the message remains unchanged and is labeled.

Example 4 - Issue a request: suppose agent A_i received two messages: *abc* and *ac*, then agent could issue a request for *b* if *b* is a necessary part for building the final model.

3.2.4 Incorporating neighbours' descriptions

An agent has to decide how to incorporate neighbours's descriptions when those descriptions arrive after some delay. The weighted average aggregation function [18, 19, 19, 20, 27] is suitable for the case where the number of clusters $\boldsymbol{k}$ is fixed to be the same for all agents only. Moreover, it assumes that the underlying distribution of measurement to be weighted is uniform.

[1]Note that agents will do repeat clustering if reading before transmitting is applied, since it always find new message in the inbox after clustering. This will drain the energy. Alternatively, agents could do only one around of clustering after it receives all information



Example 2. Given three clusters that generated from uniform distributions in a 2D space (figure on the left), K-means forms three cluster areas as balls centered at their gravity center (figure in the middle). However, the balls are not fit well with the data points. It could be further improved by extra shape information of the cluster (figure on the right).

Thus, a data aggregation function with adaptive number of clusters is desired.

We proposed a generation based method to integrate message from neighbours. When we receive descriptive summaries, we generate new data points by sampling from those descriptions, and we sample in proportion to the cluster counts. The sampled data is then combined with the local data, and provides the input to next round of local clustering. For the PCA/bounding box description, we generate points uniformly at random in the 40% box, and then generate points uniformly in each sector to extend to the 80% box, and then repeat to fill the remainder of the 100% box (Fig 5, for the same case as Fig 4). For the Gaussian summaries, we simply sample from the Gaussian distribution.

Algorithm 3 shows the re-generation steps for three clustering variants. If K-means is used as clustering algorithm, nested bounding boxes, density and centroids are used to describe the clusters. Then uniformly generate the same number of data points inside the nested bounding boxes. If GMM or SG is used as clustering algorithm, simply regenerate the data points based on the mean and standard deviation of clusters. The performance of this regeneration method is evaluated in Section 5.

Algorithm 3: Regeneration algorithm

Input: Summary descriptions $S = \{S_1, S_2, \cdots, S_k\}$;
Output: Regenerated dataset R_s;
for *each cluster description* $S_i \in S$ **do**
  if *Clustering algorithm is K-means* **then**
    /* p: array of box percentile, ;
    s_{val}: corresponding values of p */
    $S_i = \{n, p, s_{val}\}$;
    Generate data points;
    Append generated dataset to R_s;
  else if *Clustering algorithm is GMM or SG* **then**
    /* μ: mean of cluster,;
    val: the variance of cluster */
    $S_i = \{n, \mu, var\}$;
    Randomly generate n data points based on μ *and val;*
    Append generated dataset to R_s;

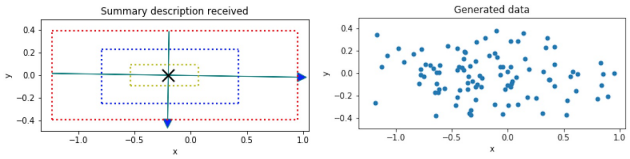


Fig. 5: Bounding boxes and the corresponding regenerated data

3.3 Agent knowledge: known number of agents

We consider two different scenarios for the initial knowledge agents have of the rest of the network. In this first scenario, we assume that each agent in the network knows the total number of other agents. In any uncontrolled process, we run the risk of creating a feedback problem, where the neighbours of an agent send it summaries which already incorporate that agent's own cluster descriptions. For example, A sends its summaries to B, which incorporates the data, generates new summaries, and sends them on to C; C updates its global view, and passes it on to A; if A then incorporates that model into its local data, it will effectively give double the weight to its own data. To avoid this problem, each agent can simply transmit its own summaries labeled with its own ID, and then simply relay other received previously unseen summaries without re-clustering. Once it has received the correct number of summaries, it can combine them all to get a global picture. At that point, it can transmit the global summary, with a flag to indicate that no new information will be received. Each agent receiving this termination message stops its own process, adopts the global summary, and retransmits the message. We call this algorithm variant Model Merge after Filtering 1 (MMF1), shown in Algorithm 4. We note that there is some privacy loss here, since at least one agent will see the individual summaries from each other agent.

Compared to methods in [17, 19], our proposed algorithms are asynchronous, which means agents do not have to wait for message from all agents to proceed to the next step. It could avoid the performance degradation caused by agent failure or stragglers. Further more, we provide more information about the cluster. The clustering for each iteration is more accurate, leading a higher final clustering accuracy and fewer message transmission.

3.4 Agent knowledge: direct neighbourhood only

In this scenario, we assume that each agent knows only its immediate neighbours, and does not know the identity of other agents, or even the size of the network. The same feedback

Algorithm 4: MMF1 for agent N_i in n agent network

```
Require: Local dataset X_i;
Input  : Message box B
Output: Final representatives of clusters
Randomly generate k initial centroids in the space;
Clustering X_i with specific centroids as input;
Initialise list of received model T;
while not terminated do
    if Message box B is not empty then
        if Receive terminate message then
            Call algorithm 3;
            Send terminate message to all neighbours;
            Terminated;
        else if Receive data message x and x ∉ T then
            Add x to T;
            if len(T) == n then
                Compute global view;
                Send converge command to neighbours;
            else
                Share message x with neighbours;
```

problem as in the previous scenario still applies. In this case, we cannot wait until all summaries are received, since no agent knows how many summaries are expected. Instead, we exploit the IDs that we can attach to each model. When an agent receives a summary constructed only from IDs it has not seen before, it applies the basic summary incorporation approach. If it receives a summary constructed entirely from IDs it has already seen, it ignores it. If it receives a summary built from some old and some new IDs, it then applies a model subtraction procedure, to avoid the feedback problem. It generates sample data points for the parent model, then generates temporary data points for the model to be subtracted, and for each temporary data point, it removes the closest data point generated for the parent model. If it does not have an appropriate set of smaller models to subtract, it identifies a small missing set, and sends a request to one or more neighbours which transmitted a superset. Those neighbours either reply with the summaries, or issue their own requests in turn. The algorithm, MMF2, is described in Algorithm 5.

Algorithm 5: MMF2 for agent N_i

```
Require: Local dataset X_i ;
Input  : Message box B
Output: Final representatives of clusters
Randomly generate k initial centroids in the space;
Clustering X_i with specific centroids as input;
Sending representatives and model name M_i to
  neighbours;
while not terminated do
    if Message box B is not empty then
        if Received message then
            Call algorithm 3;
    else
        turn to sleep but ready to work;
```

Algorithm 6 shows the procedure for model subtraction. First, the final model $\boldsymbol{f_{model}}$, which contains all unique message it saw, is computed and the largest subset $\boldsymbol{l_{sub}}$ of $\boldsymbol{f_{model}}$ is computed as well. For instance, agent i received two messages: abc and $bcef$, then the final model $\boldsymbol{f_{model}}$ is $abcef$ and $\boldsymbol{l_{sub}}$ is $bcef$. After that, $\boldsymbol{d_{set}}$ is defined as the difference between $\boldsymbol{l_{sub}}$ and $\boldsymbol{f_{model}}$. Then, the elements of $\boldsymbol{d_{set}}$ are searched in History table $\boldsymbol{T}$. If any item is still missing, send out a request to specific neighbours who have transmitted this message before. Finally, models for $\boldsymbol{f_{model}}$ are combined by substracting and regenerating and the regenerated dataset is clustered to get the new model.

Algorithm 6: Model Subtraction

```
Input  : History table T̄, Message list m_i (0 ≤ i ≤ N)
Output: regenerated dataset R_s
H ← set of agent IDs in T̄;
f_model ← union of H & agent IDs in m_i;
T̄ = T̄ + m_i;
if there exists a set of model IDs S ∈ T̄ such that S = H
  then
    f_model = S
else
    l_sub ← biggest subset from T̄;
    d_set ← f_model \ l_sub;
    Find elements of d_set missing from T̄;
    Send request to neighbours to get those models;
    f_model = l_sub + d_set;
Call algorithm 3 with f_model as input;
Clustering regenerated dataset;
Summarise the clusters;
Return new model of clusters;
```

As mentioned before, we consider two different assumptions: all agents may finish with slightly different models or must finish with identical models. The basic process for these two assumptions is as follows:

1) If we accept that agents could end up with slightly different models, then the agent picks the model that is received or produced first as their final model and ignores any model that comes later.
2) If converging with identical models is required, all agents choose the model with the earliest timestamp as their final model. In this case, an agent will update its model only when the received model has an earlier timestamp than its previous model.

Example 5: suppose there are three agents 0, 1, 2 in a chain, where agent 0 connects to the other 2 agents, and the clustering algorithm used is K-means. Suppose the transmission delay for the first round of communication is T_{01} = 0.2s, T_{02} = 0.3s, T_{10} = 0.4s and T_{20} = 0.1s, where T_{20} means the delay for transmission from agent 2 to agent 0. We assume that agents in this network do not know the size of network, the $\boldsymbol{k}$ value is not pre-defined and agents may finish with slightly different models. The *basic steps* is as follows:

- All agent will run one-round K-means with some initial centroids, and compute the initial model (contains centroids, counts and cluster description). The timestamp for each agent after initialization is [0.76, 0.85, 0.75].
- Agent 0 receives message from agent 2 at timestamp 0.85s. Then it regenerates the data points based on this message.

These regenerated data points will be combined with its local raw data points. Agents 0 computes new centroids by doing K-means on the combined data points. The difference before and after updating is substantial, so agent 0 does clustering again (finish at 1.87s). After that, agent 0 sends out the updated model to its neighbours agent 1 and 2.

- Similarly, agent 1 receives message from agent 0 at timestamp 0.96s and finishes clustering at timestamp 1.58s. Again, the updated model is sent to its neighbors agent 0. Agent 2 receives message from agent 0 at timestamp 1.06s. It finishes processing at timestamp 1.6s and shares its model with agent 0.
- Agent 0 receives message from agent 1 at timestamp 1.25s. At this time, agent 0 is doing clustering (finish at 1.87s), it will read this message after the processing.
- Agents 0 read its inbox at 1.87s and recomputes the centroids. There is only slightly difference before and after updating, so it stops updating. It sends out a convergence message (the global model is included) to agent 1 and 2.
- Agent 1 and 2 will accept the global model sent from agent 0 as its final model.

4 Time Complexity

Algorithm 4 and 5 consists of updating local model, generating data points and computing the global models. The time complexity to generate N data points uniformly or from a normal distribution is bounded by $O(N)$. First, we consider the scenario that the number of clusters $\boldsymbol{k}$ is pre-defined. If the size is known in advance, an agent computes the global model as soon as it receives local model descriptions for each agent in the network. In this case, the time complexity depends on the clustering algorithm used to compute the local and global models.

1) K-means is used as clustering algorithm. Since we calculate the distance in a 2-dimensional space, we assume the cost of computing the distance from a point to a cluster centroid is constant. Running t iterations of K-means loops (Lloyd's algorithm) exhibits the complexity of $O(N\boldsymbol{k}pt)$, where $\boldsymbol{k}$ is the number of clusters, p is the dimensionality of the data [38]. Usually the iterations t and $\boldsymbol{k}$ are small, so the time complexity is $O(N)$ [2] [39]. In this case, PCA is used to compute the nested bounding boxes. Computing PCA involves a time complexity of $O(p^2 * h + p^2 * N)$ [40], where p is the dimension of the data and h is the number of eigenvector in a reduced dimension to be computed [3]. Given the underlying dataset is 2-dimensional, computing the PCA takes time in $O(N)$. So the overall time complexity is bounded by $O(N + N + N + N) = O(N)$.
2) GMM is used as clustering algorithm. The time complexity of GMM is $O(N * \boldsymbol{k} * p^3) = O(N)$ [42]. The overall time complexity is bounded by $O(N + N + N) = O(N)$.
3) SG is used as clustering algorithm. Compared the process with that of K-means, the computation of PCA is not required. Applying a Gaussian process to a dataset of size N has complexity $O(N^3)$, and it could be reduced to $O(N)$ by various approximate techniques (i.e., partition the dataset into several groups) [43]. Since we fit each cluster to a separate Gaussian, so the overall complexity is $O(\boldsymbol{k}\bar{N})$, where $\bar{N}$ is the size of cluster and $\bar{N} < N$. So the overall time complexity is bounded by $O(N + \boldsymbol{k}\bar{N} + N) = O(N)$.

When an agent is only aware of its immediate neighbours, the agent has to compute the provisional global model whenever received new message. Suppose an agent has to update T rounds until convergence.

1) K-means is used as clustering algorithm. In this case, an agent has to compute PCA, re-generate data points and do clustering for T rounds. So the overall time complexity is bounded by $O(N + T(2N + N)) = O(TN)$.
2) GMM is used as clustering algorithm. The overall time complexity is bounded by $O(N + 2TN) = O(TN)$.
3) SG is used as clustering algorithm. The overall time complexity is bounded by $O(N + 2TN)) = O(TN)$.

When the number of cluster $\boldsymbol{k}$ is not pre-defined, an extra step to define $\boldsymbol{k}$ is required. The time complexity of silhouette method to define is bounded by $O(N^2)$. The time complexity for the distributed algorithms under this scenario, no matter which clustering algorithm is used, is bounded by $O(N^2)$. If agent is only aware of its neighbours, the time complexity is bounded by $O(N^2 + 2Tn)) = O(N^2)$.

The centralised method C_R consists of selecting the number of clusters $\boldsymbol{k}$ and doing a complete clustering. The overall time complexity for three clustering variants is bounded by $O(N^2 + N) = O(N^2)$. The centralised method C_B takes three steps: selecting the number of clusters $\boldsymbol{k}$, generating data points based on received local models, and computing the final model and it takes time in $O(N^2 + N + N) = O(N^2)$.

5 Evaluation of nested bounding boxes

In this section, we evaluate the performance of proposed method to describe the clusters with nested bounding boxes before showing the results of proposed algorithm (see Section 6). The performance is measured by comparing the summary description of generated data points with the ground truth summary description. To measure how close the new summary description is to the ground truth summary description, we calculate the difference of three measurements before and after regeneration:

1) E_d: Euclidean Distance between the original centroids and new centroids,
2) E_s: shift distance of bounding box (measures the shift distance of four coordinates (leftmost, rightmost, bottom, top) of the box), and
3) E_r: rotation degree from old PCs to new PCs.

We evaluate the performance with different underlying distributions, different percentiles and different sizes of datasets. Experiment results show that the summary descriptions do appear to be stable after data regeneration, changing the coordinates, shape and rotation by only small amounts. And we could get a similar new generated dataset even the size is small. Moreover, we observe that percentiles, which describe more information about the boundary points (located inside the 100% bounding box but outside the second inner bounding boxes), performs the best. Note that there appears to be little advantage in generating more bounding boxes in the summaries. More details about the experiments are attached in the technique report (https://ershao.github.io/Response)

Possible issues caused by regeneration. Regenerating data from the bounding box descriptions does seem to be feasible, and would allow an agent to generate similar clusters of data. But

[2] In the worst case, k-means is exponential.

[3] When p is larger than N, the time complexity is bounded by $O(min(p^3, N^3))$ [41]

there may be an issue if this is done repeatedly. If a bounding box extends to exactly the bounds of the extreme points, when we regenerate points inside the box, we are unlikely to generate points on the edges, and so the spread of the cluster shrinks after each regeneration, and repeated regeneration will make it worse. In order to address this issue, we considered four potential solutions:

1) P_1: do nothing.
2) P_2: impose 4 data points on the extremes of the outer box during regeneration.
3) P_3: adjust the bounding boxes on creation by expanding the inner boxes to midway to the next relevant point.
4) P_4: combine step 3 and 4 above.

Figure 6 shows that the proposed algorithm without any make-up plan (P_1) and with bounding line extension outperforms other schemes (P_3). Comparing to P_1, there is little difference in convergence time, but the moving distance of three bounding boxes and rotation increase, and accuracy decreases when we make up the shrink of outer box only (P_2). If we extend the inner boxes to midway to the next relevant point (P_3), the convergence time decreases. Although there is slightly increase in rotation (but with a higher standard deviation), the accuracy still remains high. When step 3 and 4 combined (P_4), the rotation increases with a higher standard deviation and accuracy decrease. The most possible reason lies in the re-generated data. Without any shrink make-up plan, data points of different clusters are tender to be more separated from each other, that is the reason why the clustering accuracy is high by this scheme.

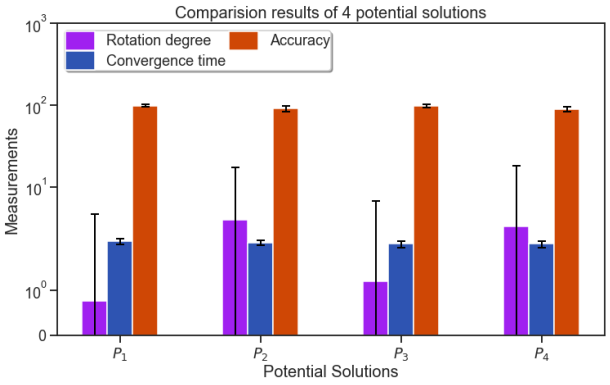


Fig. 6: Comparision results of different solutions with elliptical well separated clusters

However, when there are overlaps among clusters, the later scheme outperforms the previous scheme (see Figure 7). In some practical applications, there are slight overlaps, even huge overlaps between clusters, we used the scheme that extending the bounding line to half way to the closet points to address the shrink up problem.

6 Empirical Evaluation

To simulate the operation of the algorithms on a WMN, we implement an Asynchronous Message Delay Simulator, based on [44]. Random network topologies are generated based on [45], where the probability p_N is larger than $\frac{(1+\varepsilon)ln(N)}{N}$, where ε is a positive constant, to ensure that a random graph generated is a connected graph (we set $\varepsilon = 1$). To generate dense graph topologies, p_N is set to 0.8. The initial raw data is generated in

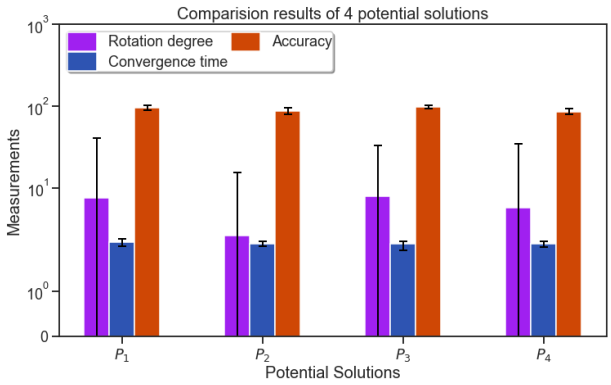


Fig. 7: Comparision results of different solutions with overlapped clusters

k clusters, sampled from a two-dimensional normal distribution, scaled to the range [0,1] before being assigned randomly to network agents. Each message transmission delay is generated uniformly from the range [0.5, 1.0] [19], and agents begin their work at time randomly selected in [0, 0.1s].

In addition to the convergence time and communication costs, we also measure the number of learned clusters for all agents on average. The accuracy measurement percentage of membership mismatch (PMM) used in [17, 22] measures how accurate each agent's model is on its own data compared to the centralised baseline or to the ground truth (original generation of the data points), and it is denoted by M1. Before we distribute the data points over the agents, we run a single centralised clustering method, to get a benchmark clustering result. We use PMM as our formal measure of clustering error.

$$PMM^{(i)} = 100\frac{|\vec{x} \in X^{(i)} : L_c(\vec{x}) \neq L_p^i(\vec{x})|}{|X^{(i)}|} \quad (2)$$

where $L_c(\vec{x})$ denotes the label of the cluster to which $\vec{x}$ is assigned at the end of centralized clustering and $L_p^i(\vec{x})$ denotes the label of the cluster to which $\vec{x}$ is assigned once the agent reaches the termination state. However, this penalises the agent when its local data does not fit well with global model learned. In this paper, we propose that the accuracy against a new test dataset is applied (denoted by M2). Measuring the accuracy against all raw data is expensive and slow, so we use a new test dataset that sampled from the same ground truth distribution instead.

Two types of underlying network topology are used: dense and sparse graphs. Three different types of datasets are tested: fully symmetric multi-dimensional Gaussians, partially asymmetric multi-dimensional Gaussians, and rectangle uniformly distributed data. For each type of dataset, there are five generated clusters and three different types of clusters: heavily-overlapped clusters, slightly-overlapped clusters and well-separated clusters. Figure 8 shows an example of different overlaps between cluster.

We evaluate a number of different instantiations of the AAC framework. MMFi-P uses full k-means, exchanging PCA and Bounding Box summaries, regenerating data by sampling, for the two MMF variants. Similarly, MMFi-SG uses full k-means, but then exchanges separate Gaussian models with counts fitted to each cluster. MMFi-GMM fits a Gaussian Mixture Model for clustering, and exchanges the Gaussian models with counts. We also evaluate [17] partial k-means, exchanging centroids and

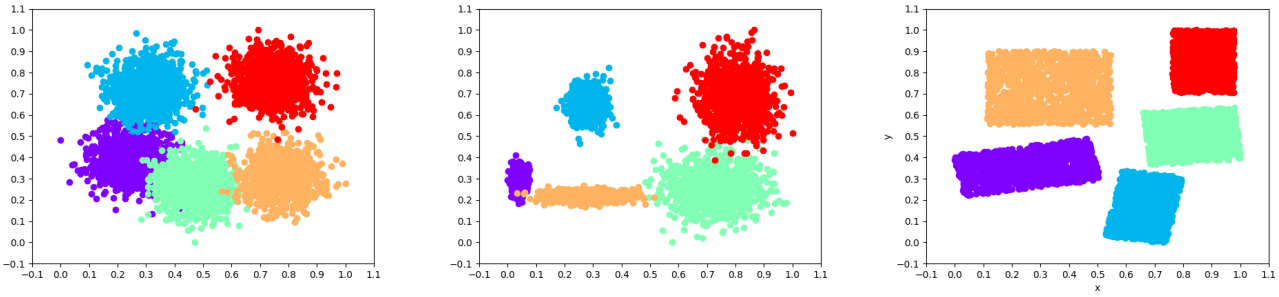


Fig. 8: An example of different overlaps between clusters: fully symmetric Gaussians with heavy overlap, partially symmetric Gaussians with slight overlap and well separated uniform distributions (left to right)

counts with all neighbours, and [19, 20] (full k-means, exchanging weighted centroids and counts with a single neighbour).

Since GMM is a soft clustering method, which assigns to a single data point a probability of membership of each cluster, PMM may be unfair to GMM, since it can never achieve 100% clustering accuracy. In order to compare with the hard assignment obtained by k-means, we assume that the final assignment of a data point to a cluster is determined by the highest probability. In addition to clustering accuracy, we measure the convergence time, which records the interval between the time when the first agent initialised and the last agent terminated. For communication costs, we record the total number of received messages. The notation is summarised in Table 1.

None of our methods use a central controller, and we assume an additional process which detects termination if required. For termination time, we record the last time at which any agent or message was active. For MMF2, we also measured the time that the first agent reached stability (t_0), and the time that the last agent reached stability (t_1). In those MMF2 experiments, t_2 is the final termination.

S1_A1	The size of network is known and agents may finish with different models
S1_A2	The size of network is known and agents must finish with identical model
S2	Agents are only aware their neighbours and may finish with different models
C_R	Centralise all raw data point to a central agent
C_B	Centralise basic models to a central agent
$Accu\_c$	Accuracy against centralised clustering method
$Accu\_g$	Accuracy against ground truth
$M1$	Evaluate the performance on agent's own data only
$M2$	Evaluate the performance on a new test data

TABLE 1: Notation of different methods and measurements

The number of messages received is not necessarily sufficient to measure communication cost, since large messages require more packets and thus longer transmission times. For IEEE 802.11ac, the maximum size of a single packet (MPDU) is 11,454 bytes. For our experiments, we limit the number of clusters to 5 (ground truth). Five centroid locations plus counts, or five Gaussians plus counts, can be described in just over 1000 bytes. The principal components and bounding boxes for five clusters can be described in just 9,272 bytes, and thus, for our experiments, each model description can be transmitted in a single packet, and so the total number of messages is a good proxy for communication energy costs. In our experiments, each agent is randomly assigned 200 raw data points, and the size is 22,500 bytes. So there are in total 3 packets to send if centralising raw data to a central agent is applied.

6.1 The number of cluster k is known

In this section, we show the comparison result when the number of clusters $\boldsymbol{k}$ is known to all agents in advance. First we evaluate performance under scenario 1, in which agents know the size of the network. The results for densely connected networks of 10 agents are shown in Figure 9. MMF1-P and -SG show the highest accuracy (relative to centralised k-means); Both of these methods use full k-means at each agent to generate the initial clustering and the final clustering, but differ in the summaries that are exchanged. MMF1-P requires the fewest messages, but takes significantly longer than MMF1-SG to terminate – this appears to be because of the extra time required to analyse the clusters using PCA and to generate the bounding boxes. MMF1-GMM requires the least time to stabilise, and achieves reasonably high accuracy. The existings methods from [17], [19] and [20] are outperformed on all measures. In Figure 10, we show the result for sparsely connected networks of 10 agents. MMF-1 achieves the highest accuracy and requires the least amount of messages to converge. Again, MMF1-GMM converges faster than other methods and achieves high accuracy. Again, [17], [19] and [20] are outperformed on all measures. To summarise, selecting between MMF1-P, -SG and -GMM will depend on the relative costs of inaccuracy, message transmission, and elapsed time.

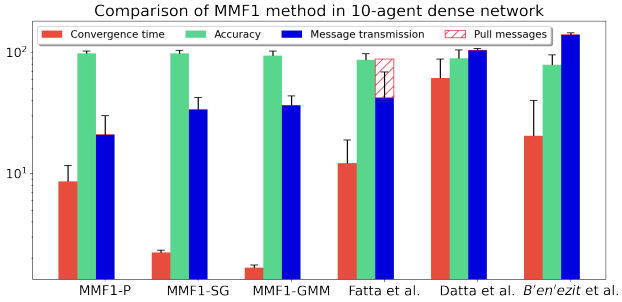


Fig. 9: Comparison of MMF1 method in 10-agent dense network.

In Figure 11 we show the result of densely connected networks for scenario 2, in which agents only know of the existence of their immediate neighbours. We report three different time measures. MMF2-P again requires the fewest messages. MMF-SG again has the highest accuracy. The relative performance of MMF2-GMM has improved, with close to the highest accuracy, and clearly the

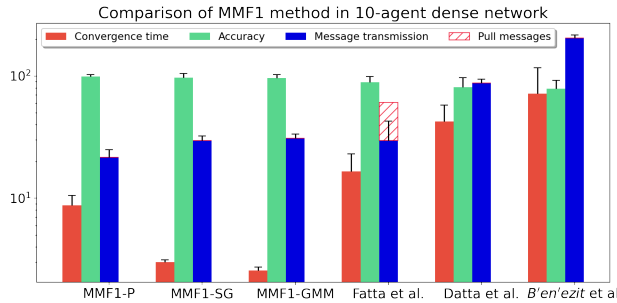


Fig. 10: Comparison of MMF1 method in 10-agent sparse network

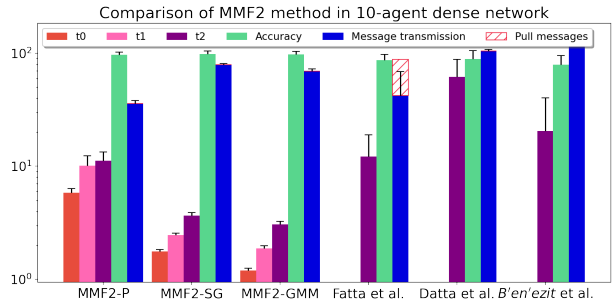


Fig. 11: Comparison of MMF2 method in 10-agent dense network

fastest termination time. [17] is still outperformed by the other methods, although the number of messages (including both data messages and polling messages) are no longer significantly higher. [19] and [20] are still outperformed on all measures. The result for sparse networks are shown in Figure 12. MMF2-SG shows the highest accuracy and MMF2-P requires the fewest messages but takes substantially longer time than MMF2-SG to terminate. MMF2-GMM converges much faster than other methods and still achieves high accuracy. The methods in [17], [19] and [20] are outperformed on all measures.

Table 2 shows the experimental results on real-world datasets Flame [46], Cassini [47] and Aggregations [48]. We compare the accuracy against ground truth (denoted by A_t) as well as accuracy against centralised method (A_c), since the ground truth is now available. In term of accuracy against centralised method A_c, the proposed methods are outperformed on all datasets. However, the accuracy against ground truth A_t obtained by proposed methods is higher than the methods in [17], [19] and [20] on all datasets. More information about the clusters are described in the proposed methods, leading a better performance than the state-of-the-arts. Although the clusters are well separated in Flame [46], the structures make it a challenge to cluster. MMF1-P performs better than other methods since an additional nested bounding box is used to describe the clusters. When the underlying distributions are normal or there are tiny gaps between clusters [47, 48], GMM achieves a much higher accuracy than other algorithms.

TABLE 2: Comparison of performance on different datasets

Datasets		*Flame*	*Cassini*	*Aggregation*
MMF1-P	A_c	93.25±3.67	72.68±19.0	88.93±4.91
	A_t	**89.4**±1.54	66.45±0.56	85.25±3.16
MMF1-SG	A_c	93.71±4.01	76.92±18.9	88.8±4.60
	A_t	89.61±2.67	66.1±4.24	84.85±3.61
MMF1-GMM	A_c	90.26±6.14	82.15±7.47	87.3±4.60
	A_t	87.1±2.42	**82.25**±4.24	**87.86**±4.19
Fatta	A_c	**98.45**±3.03	**94.7**±7.88	**95.03**±5.63
	A_t	88.83±3.25	66.7±.1.02	81.83±4.57
Datta	A_c	94.68±4.9	76.02±17.2	89.08±5.02
	A_t	85.85±1.35	67.5±2.5	84.83±3.16
Bénézit	A_c	95.4±6.0	82.5±2.0	88.35±4.32
	A_t	84.25±3.0	72.5±3.2	80.29±3.20

Table 3 shows the significance study of the experimental results. A statistical comparison is thought to be significant when its P-value is less than 0.05. We apply the Tukey test to conduct pairwise comparisons of the approaches. The statistical analysis is carried out through the built-in functions in Python. It shows that, for all the scenarios, the null hypothesis is rejected with the

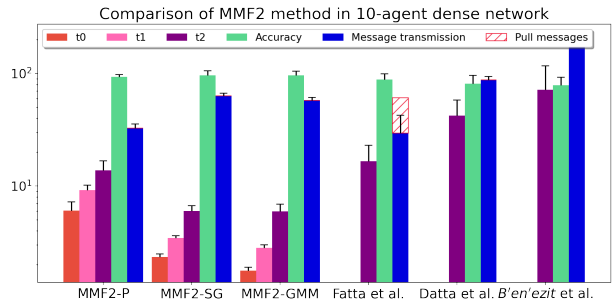


Fig. 12: Comparison of MMF2 method in 10-agent sparse network

P-values all less than 0.05. It indicates that there is a significant difference between proposed algorithms and the state-of-the-arts and the improvement is significant.

Then we show the results with different size of networks. Note that we only compare the proposed algorithm AAC, where K-means is used as clustering algorithm and nested bounding boxes are used to describe the clusters, with the existing methods. Figure 13, 14 and 15 shows the performance on dense network. When agents in the network are well connected, the accuracy decreases when the size of network increases. But still the accuracy is above 80%, and the highest accuracy is achieved by AAC. In term of convergence time, method in [17] and AAC converge faster than other methods, and they are robust even in large scale network. Figure 28 shows the network overload. The least amount of network overhead is obtained by AAC, followed by method in [19].

Figure 16, 17 and 18 show the accuracy, convergence time and transmitted message in the sparse network, respectively. The highest accuracy is achieved by method in [17], followed by AAC. Similarly, method in [17] and AAC are the farthest method. Again, method proposed by [19]. and AAC require the least amount of message transmission. When the size of network is less than 50, AAC performs slightly better. When the size of network is large than 50, method proposed in [19] is slightly better. The most possible reason behind this is that, the gossip way shows some advantages to broadcast information in a sparse network.

In summary, the methods that use full clustering at each agent on each cycle, and which exchange more informative descriptions, outperform [17]. Taking MMF2-GMM as representative, it achieves 10 percentage points higher accuracy relative to centralised k-means (reduces the misclassification rate from 12% to 1.7%), reduces the message count by 20%, and reduces elapsed time by 75%. In all cases, we respect the privacy of the original data, and do not exchange any individually identifiable data points.

TABLE 3: Pairwise Tukey test for proposed methods and the state-of-the-arts. ⋄ denotes the difference of mean.

	Dense Graph			Spare Graph		
MMF1	*Time* (⋄, P value)	*Accuracy* (⋄, P value)	*R-msg* (⋄, P value)	*Time* (⋄, P value)	*Accuracy* (⋄, P value)	*R-msg* (⋄, P value)
P – Datta	(-3.6, 0.001)	(11.30, 0.001)	(-67.80, 0.001)	(-8.0, 0.001)	(10.20, 0.001)	(-38.50, 0.001)
P – Fatta	(-53.0, 0.001)	(9.20, 0.001)	(-84.0, 0.001)	(-33.6, 0.001)	(17.5, 0.001)	(-66.0, 0.001)
P – Bénézit	(-12.0, 0.001)	(20.1, 0.001)	(-119.67, 0.001)	(-63.0, 0.001)	(20.4, 0.001)	(-183.1, 0.001)
SG – Datta	(-10.0, 0.001)	(11.5, 0.001)	(-54.0, 0.001)	(-13.6, 0.001)	(8.40, 0.001)	(-31.0, 0.001)
SG – Fatta	(-59.0, 0.001)	(9.20, 0.001)	(-71.0, 0.001)	(-39.3, 0.001)	(15.70, 0.001)	(58.20, 0.001)
SG – Bénézit	(-18.3, 0.001)	(16.3, 0.001)	(-18.56, 0.001)	(-68.30, 0.001)	(18.60, 0.001)	(-185.2, 0.001)
GMM – Datta	(-10.6, 0.001)	(7.40, 0.001)	(-19.0,0.001)	(-14.0, 0.001)	(7.70, 0.001)	(-30.0, 0.001)
GMM – Fatta	(-60.6, 0.001)	(5.20, 0.001)	(-68.1, 0.001)	(-39.60, 0.001)	(15.0, 0.001)	(-67.3, 0.001)
GMM – Bénézit	(-19.0, 0.001)	(15.30, 0.001)	(-103.0, 0.001)	(-69.03, 0.001)	(17.8, 0.001)	(-174.1, 0.001)

	Dense Graph			Spare Graph		
MMF2	*Time* (⋄, P value)	*Accuracy* (⋄, P value)	*R-msg* (⋄, P value)	*Time* (⋄, P value)	*Accuracy* (⋄, P value)	*R-msg* (⋄, P value)
P – Datta	(-1.13, 0.001)	(9.7, 0.001)	(-52.0,0.001)	(-3.16, 0.001)	(4.50, 0.001)	(-27.3, 0.001)
P – Fatta	(-50.50, 0.001)	(7.6, 0.001)	(-68.30, 0.001)	(-29.40, 0.001)	(11.8, 0.001)	(-137.5, 0.001)
P – Bénézit	(-8.8, 0.001)	(18.9, 0.001)	(-104.70, 0.001)	(-58.10, 0.001)	(14.60, 0.001)	(-172.54, 0.001)
SG – Datta	(-9.30, 0.001)	(11.19, 0.001)	(-8.20,0.001)	(-10.7, 0.001)	(8.02, 0.001)	(3.30, 0.001)
SG – Fatta	(-58.0, 0.001)	(9.1, 0.001)	(-45.20, 0.001)	(36.6, 0.001)	(15.36, 0.001)	(-24.20, 0.001)
SG – Bénézit	(-17.0, 0.001)	(20.30, 0.001)	(-60.0, 0.001)	(-65.70, 0.001)	(8.02, 0.001)	(-141.10, 0.001)
GMM – Datta	(-8.90, 0.001)	(20.70, 0.001)	(-18.30,0.001)	(-10.70, 0.001)	(7.60, 0.001)	(-2.20, 0.001)
GMM – Fatta	(-58.30, 0.001)	(8.60, 0.001)	(-45.40, 0.001)	(-36.30, 0.001)	(15.40, 0.001)	(-30.10, 0.001)
GMM – Bénézit	(-16.50, 0.001)	(18.80, 0.001)	(-70.50, 0.001)	(-65.60, 0.001)	(17.70, 0.001)	(-147.6, 0.001)

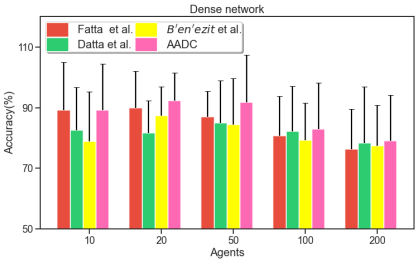


Fig. 13: Accuracy on dense network

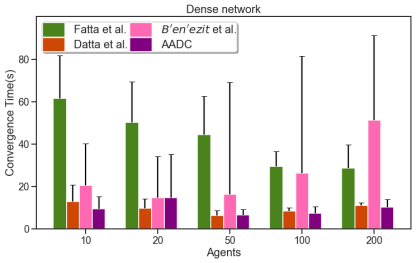


Fig. 14: Convergence on dense network

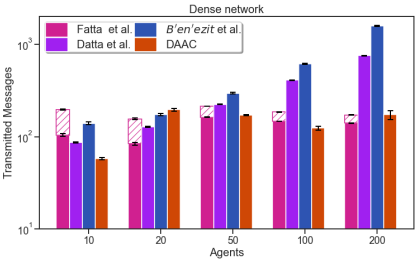


Fig. 15: Network overload on dense network

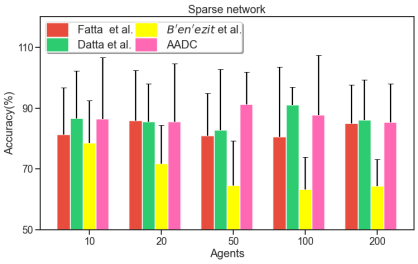


Fig. 16: Accuracy on sparse network

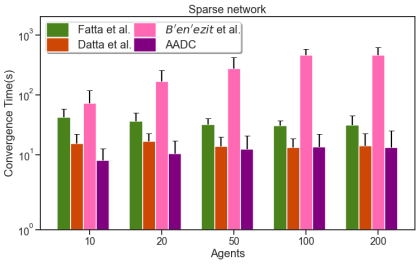


Fig. 17: Convergence on sparse network

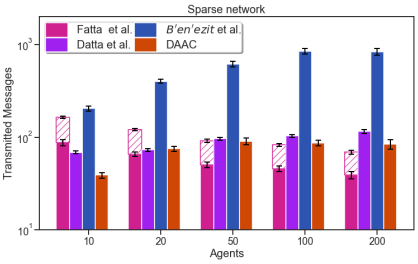


Fig. 18: Network overhead on sparse network

6.2 The number of cluster k is not known

6.2.1 K-means as clustering algorithm

First, we measure the performance when K-means is used as the local clustering algorithm. Figure 19 shows the comparison result when the underlying network is dense. Centralising raw data points C_R converges faster than other methods when inter-package delay is not considered. Centralised basic model C_B requires fewest message transmissions. Since we assume that the routing table is known in advance and a shortest-path based tree is used, we expect this method will show the least network overhead. However, the standard deviation shows that some agents relayed many messages while other agents did not.

For distributed methods, $S1\_A1$ converges faster than other methods. This is as expected because $S1\_A1$ assumes that the size of network is known and agents could end up with different models, where agents do not need to keep updating their models and sending more packets, and that fewer communications are needed to share the final models. $S2$ requires the largest amount of message transmissions, followed by $S1\_A2$. The most likely reason behind this is each agent is only aware of its immediate neighbours under $S2$ and has no idea about the size of network. Agents in S2 are repeatedly updating their models and transmitting them again, while in $S1\_A2$, if multiple final models are created independently, agents may be forced to accept updates with lower timestamps and so have to retransmit a final model to their

neighbours. The accuracy of $S2\_A2$ is similar to that of $S1\_A2$, but requires longer time to converge (around 80s in three variants of underlying datasets).

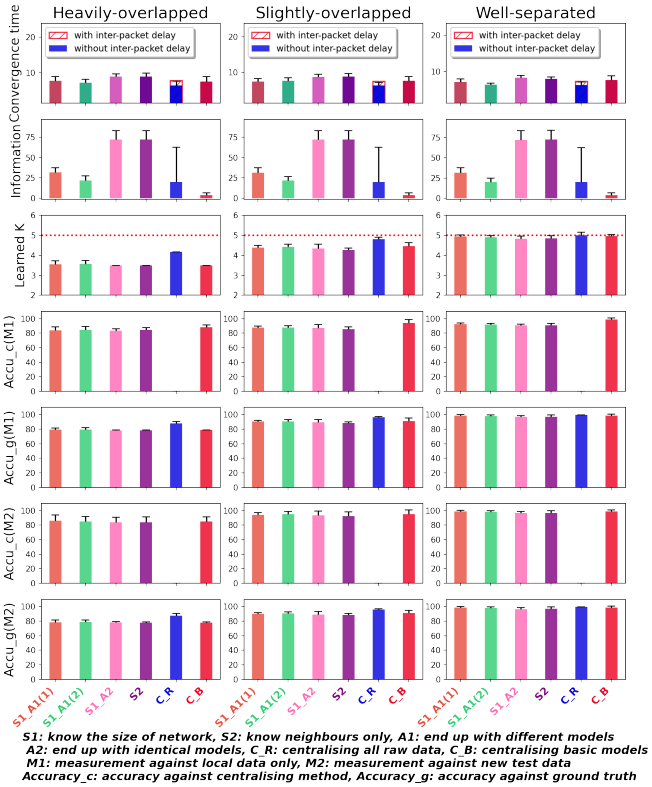


Fig. 19: K-means with symmetric multi-dimensional Gaussians on dense network

The extent of overlapping of clusters affects how accurate the number of learned clusters is, compared to the ground truth. The third row of figure 19 shows that centralised methods predict a more accurate number of clusters than the distributed methods for the overlapped cases. For the well-separated case, they are all getting approximately 5. The last four rows show the accuracy against centralised method and ground truth, measured by M1 and M2 separately. Centralising all raw data points C_R achieves the highest accuracy against ground truth. That is as expected. C_R receives all the raw data unfiltered, while the other methods are summarising, regenerating and re-clustering, and so can be expected to introduce errors.

When accuracy against the centralised method is measured with an agent's own data M1, centralising all basic models C_B achieves the highest accuracy. However distributed method $S1\_A1$ performs better than C_B when the same accuracy is measured with a new test data M2. It indicates that agent's local data may not fit the learned model very well in some cases. Note that when accuracy against ground truth is measured, there is very little difference between M1 and M2.

Figure 20 shows the performance when data points that generated from partially asymmetric Gaussians are used. Compared to that with symmetric Gaussians, the accuracy against data points and accuracy of the inferred number of clusters has dropped. However, the convergence time and total message transmission remains unchanged. Note that when the clusters are well separated from each other, there is no difference in accuracy against data points and accuracy of the inferred number of clusters. That is because K-means is able to cluster data points when clusters are well separated.

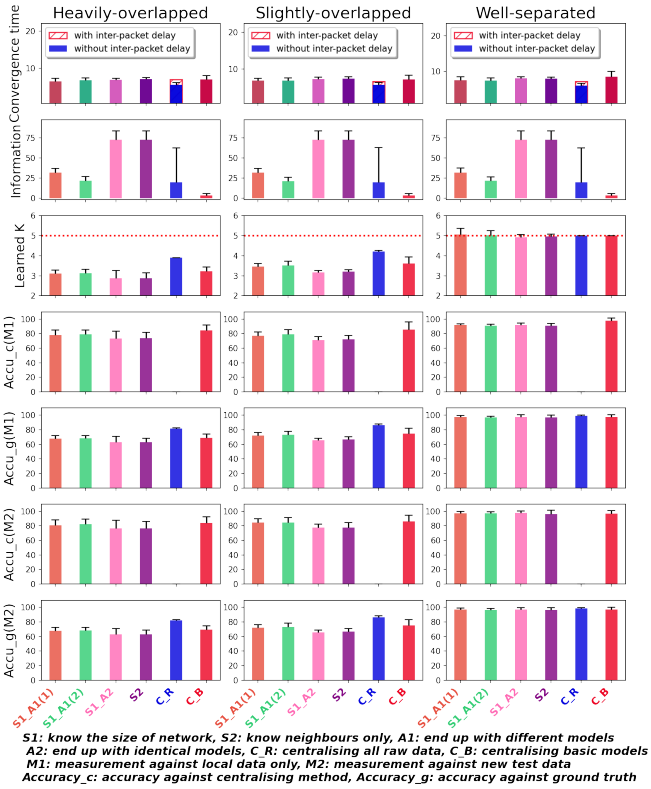


Fig. 20: K-means on dense network with asymmetric multi-dimensional Gaussians

Figure 21 shows the performance when uniformly distributed datasets are tested. Again, the convergence time and the number of transmitted messages remains stable. Compared to the previous two figures, accuracy of the inferred $\boldsymbol{k}$ is poorer. In addition, accuracy on data points against ground truth (both M1 and M2) has dropped significantly. This is as expected because it is a challenge for K-means to handle uniformly rectangle distributed data. If the data is uniform inside a long rectangle, that K-means will struggle, because points at the end of one rectangle may be closer to the centroids of another. Compared to accuracy on data points against centralised method (the fourth and sixth row), the accuracy on data points against ground truth (the fifth and seventh row) is much lower. It indicates that simply comparing to centralised method is not reliable when clustering algorithm is not fitting well with the underlying dataset.

To summarise, centralised method C_R takes the shortest time to converge if inter-packet delay is not considered, and the convergence time depends on what value you use in the simulator for inter-packet delay. Centralised method C_B requires the fewest transmissions. However, the high standard deviation shows that agents close to the root have to relay many packets while agents far away from the root only forward a few packets.

In terms of the number of learned clusters, the extent of

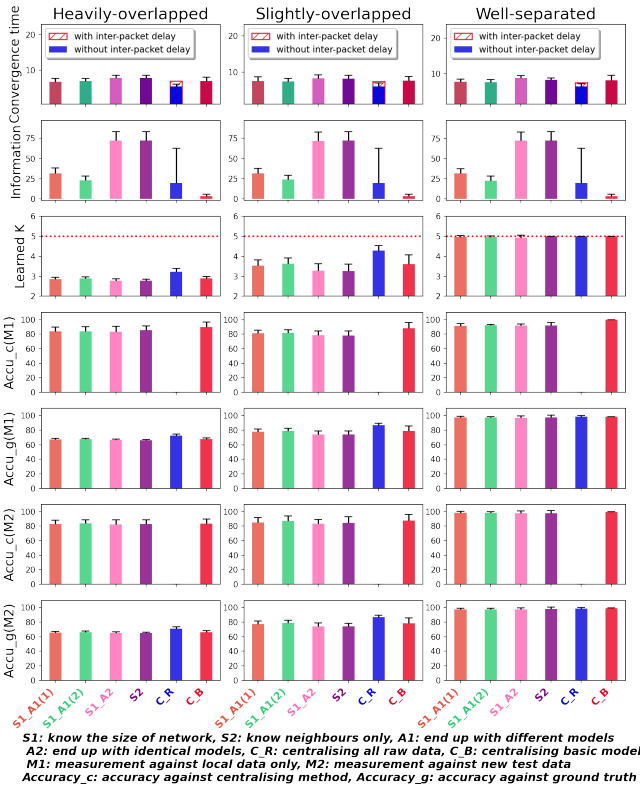


Fig. 21: K-means on dense network with uniform distributed datasets

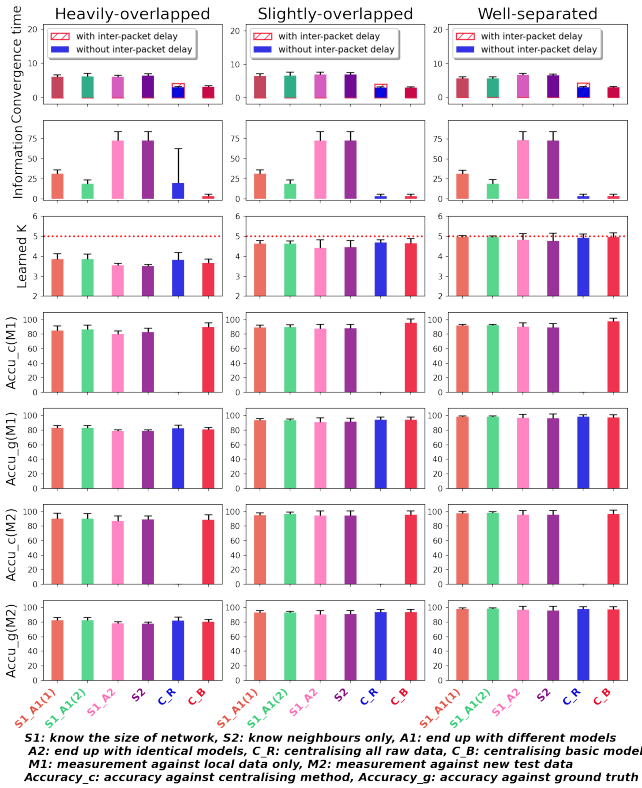


Fig. 22: GMM on dense network with symmetric multi-dimensional Gaussians

overlapping between clusters plays a vital role. When clusters are well separated from each other, our proposed algorithm matches the ground truth number of clusters perfectly, no matter what the underlying dataset and network topology are.

When K-means is used as clustering algorithm, it achieves the highest accuracy on data points against centralised algorithm and ground truth when the underlying datasets are formed from symmetric Gaussians clusters. The accuracy on data points against ground truth drops to about 70% when partially-symmetric Gaussians are used, then declines to around 65% when uniformly distributed clusters are used. However, the accuracy on data points against the centralised algorithm remains stable even when the underlying dataset appears a challenge for K-means to cluster. It indicates that the drop in accuracy is due to the underlying K-means algorithm, and not the distributed protocol.

6.2.2 GMM as clustering algorithm

In this section, we measure the performance when GMM is used as clustering algorithm [4]. In this case, the collection of clusters is described as a collection of Gaussians. Figure 22 shows the result on dense networks with data points formed from fully symmetric Gaussians. Again, centralising all data points C_R takes the shortest time to converge when inter-packet delay is ignored. In addition, C_B requires the least amount of message to transmit. All centralised methods and distributed methods achieved really high accuracy, even when there are heavy overlaps between clusters.

[4]We omit the result of SG, which is similar to that of GMM

Figure 23 shows the result on dense graphs when partially symmetric Gaussians are tested. Compared to Figure 22, there is little difference between the accuracy on data points against ground truth and centralised algorithm when clusters are well separated from each other. However, when the extent of overlapping increased, the accuracy against ground truth dropped sharply while the corresponding accuracy against centralised algorithm dropped by less than 10%.

When uniformly distributed data is used as the underlying dataset, we expected that accuracy against ground truth would be lower than that with symmetric or asymmetric Gaussians. Figure 24 shows that the accuracy against ground truth is around 62% for both M1 and M2. However, the accuracies against centralised algorithm for M1 and M2 are still high, more than 80%.

Compared to K-means and SG, the result is similar when GMM is used as local clustering method. The centralised method C_R converges slower than other methods if inter-packet delay is considered and centralised method C_B requires the fewest transmissions. When clusters are well separated from each other, our proposed algorithm learned the perfect number of clusters, achieved nearly perfect accuracy against centralised algorithm and ground truth, no matter what the underlying dataset and network topology are.

7 Conclusion and Future Directions

We proposed an asynchronous distributed clustering algorithms framework for Wireless Mesh Networks, which respect data privacy, while balancing communication cost and clustering

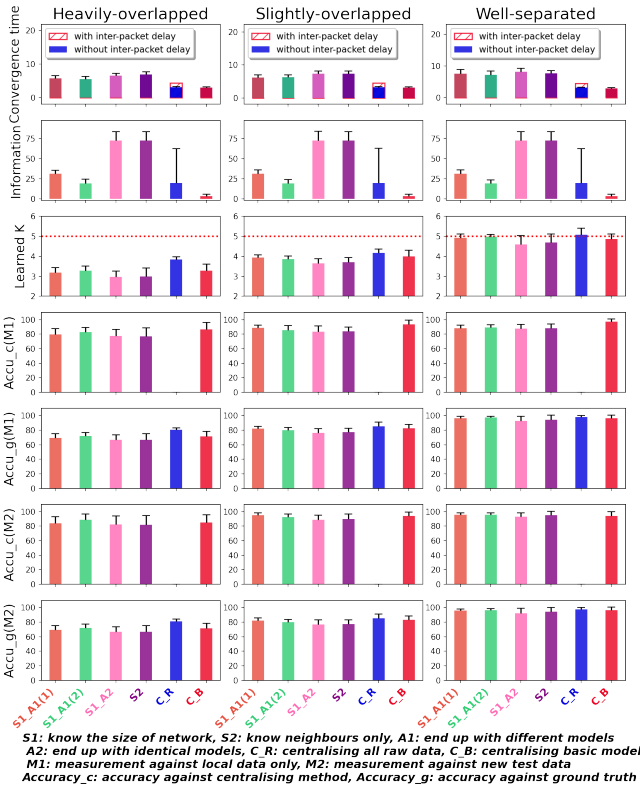


Fig. 23: GMM on dense network with asymmetric multi-dimensional Gaussians

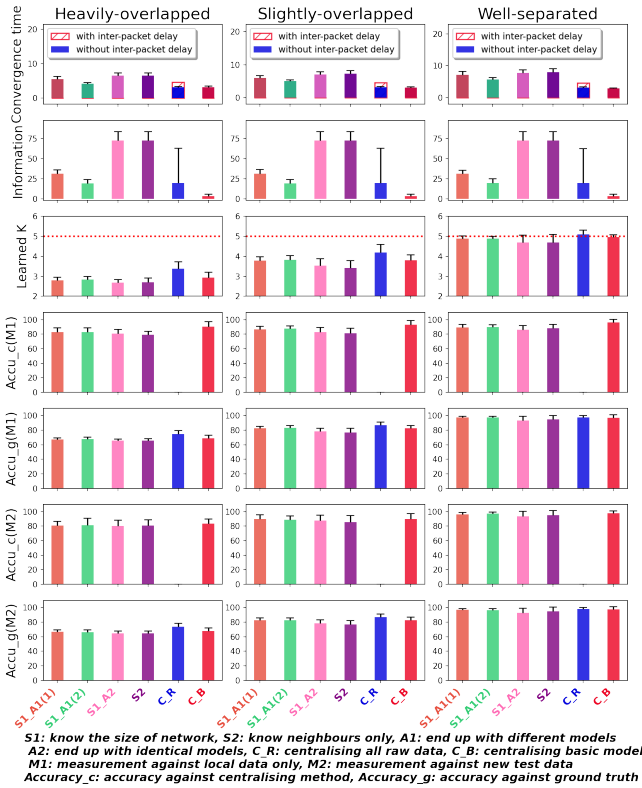


Fig. 24: GMM on dense network with uniform distributed dataset

quality. Methods that use full k-means clustering at each agent each cycle, and which exchange cluster shape and density descriptions, require fewer messages and give higher accuracy relative to centralised k-means, compared to previous methods. Distributed Gaussian Mixture Model clustering is almost as high in accuracy, and requires less elapsed time. The methods do, however, leak some privacy about the data sensed by each individual agent. The results show that more informative cluster descriptions improve distributed clustering.

More information is required to be transmitted if it is required that agents must finish with identical models. However, if we accept that agents may finish with different models, it requires much less information transmission and much less elapsed time. Agent converges fast if it knows the size of network. We expected this since an agent have to compute the provisional model whenever receives new messages if they do not know the size of network. Our proposed algorithm could learn the same number of clusters as ground truth even there is no prior information about this value.

If we assume that there is no inter-packet delay, centralising all raw data requires the transmission of less information and the shortest convergence time. However, this method suffers from the problem of single point of failure and leakage of data privacy. The standard deviation of communication cost is high, which means some agents have relayed huge amounts of information while others have not. This will drain the energy of agents and frequently recomputing the routing table will become an issue. Note that we assume that the shortest path is available for each agent in advance and we ignore the cost to compute the routing table.

The distributed method offers similar clustering accuracy to the centralised methods but relaxes the assumption that agents know the whole routing path in advance. Compared to the highest accuracy of the centralised method, the clustering accuracy achieved by the distributed method dropped by around 9% and the communication cost increased by up to 20%. But the convergence time has been reduced and the problem of a single point of failure has been avoided. In addition, we respect the data privacy.

For a more comprehensive conclusion, we will extend the evaluation, to consider networks in which agents fail (e.g. because of limited batteries). We will extend the problem to consider the case where sub-groups of agents receive data from different environments, and so the inferred cluster models should be different for each sub-group.

Acknowledgments

This research was supported by National Natural Science Foundation of China under Grant No. U20B2046, Guangdong Province Key Area RD Program of China under Grant No. 2019B010137004, Science Foundation Ireland under Grant Number SFI/12/RC/2289-P2 and 16/SP/3804 and Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme (2019).

References

[1] L. Lou, Q. Li, Z. Zhang, R. Yang, and W. He, “An iot-driven vehicle detection method based on multisource data fusion technology for smart parking management system,” *IEEE*

Internet of Things Journal, vol. 7, no. 11, pp. 11 020–11 029, 2020.

[2] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J.-S. Oh, "Semisupervised deep reinforcement learning in support of iot and smart city services," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 624–635, 2018.

[3] J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, and B. Fang, "A survey on access control in the age of internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4682–4696, 2020.

[4] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "Corrauc: A malicious bot-iot traffic detection method in iot network using machine-learning techniques," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3242–3254, 2021.

[5] H. Harb, A. Makhoul, S. Tawbi, and R. Couturier, "Comparison of different data aggregation techniques in distributed sensor networks," *IEEE Access*, vol. 5, pp. 4250–4263, 2017.

[6] Y.-Y. Ting, C.-W. Hsiao, and H.-S. Wang, "A data fusion-based fire detection system," *IEICE TRANSACTIONS on Information and Systems*, vol. 101, no. 4, pp. 977–984, 2018.

[7] Z. Ullah, "A survey on hybrid, energy efficient and distributed (heed) based energy efficient clustering protocols for wireless sensor networks," *Wireless Personal Communications*, pp. 1–29, 2020.

[8] https://www.i-scoop.eu/big-data-action-value-context/data-age-2025-datasphere/.

[9] C. Li, G. Li, and P. K. Varshney, "Decentralized federated learning via mutual knowledge transfer," *IEEE Internet of Things Journal*, 2021.

[10] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive iot networks," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4641–4654, 2020.

[11] D. K. Kotary and S. J. Nanda, "Distributed robust data clustering in wireless sensor networks using diffusion moth flame optimization," *Engineering Applications of Artificial Intelligence*, vol. 87, p. 103342, 2020.

[12] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[13] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications surveys & tutorials*, vol. 18, no. 2, pp. 1153–1176, 2015.

[14] L. Faramondi, G. Oliva, R. Setola, and C. N. Hadjicostis, "Distributed c-means clustering via broadcast-only token passing," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 1, pp. 315–325, 2020.

[15] G. B. Giannakis, Q. Ling, G. Mateos, I. D. Schizas, and H. Zhu, "Decentralized learning for wireless communications and networking," in *Splitting Methods in Communication, Imaging, Science, and Engineering*. Springer, 2016, pp. 461–497.

[16] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," *arXiv preprint arXiv:2002.10619*, 2020.

[17] S. Datta, C. Giannella, and H. Kargupta, "Approximate distributed k-means clustering over a peer-to-peer network," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 10, pp. 1372–1388, 2009.

[18] D. Dennis, T. Li, and V. Smith, "Heterogeneity for the win: One-shot federated clustering," in *ICML*, 2021.

[19] G. Di Fatta, F. Blasa, S. Cafiero, and G. Fortino, "Fault tolerant decentralised k-means clustering for asynchronous large-scale networks," *Journal of Parallel and Distributed Computing*, vol. 73, no. 3, pp. 317–329, 2013.

[20] F. Bénézit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in *2010 IEEE International Symposium on Information Theory*, 2010, pp. 1753–1757.

[21] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *ACM Computing Surveys (CSUR)*, vol. 53, no. 2, pp. 1–33, 2020.

[22] C. Qiao and K. N. Brown, "Asynchronous distributed clustering algorithm for wireless sensor networks," ser. ICMLT, 2019, p. 76–82.

[23] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.

[24] M. Khodak, M.-F. Balcan, and A. Talwalkar, "Adaptive gradient-based meta-learning methods," *arXiv preprint arXiv:1906.02717*, 2019.

[25] H. Lu, J. Li, and M. Guizani, "Secure and efficient data transmission for cluster-based wireless sensor networks," *IEEE transactions on parallel and distributed systems*, vol. 25, no. 3, pp. 750–761, 2013.

[26] R. Logambigai, S. Ganapathy, and A. Kannan, "Energy–efficient grid–based routing algorithm using intelligent fuzzy rules for wireless sensor networks," *Computers Electrical Engineering*, vol. 68, pp. 62–75, 2018.

[27] M. Bendechache, A.-K. Tari, and M.-T. Kechadi, "Parallel and distributed clustering framework for big spatial data mining," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 34, no. 6, pp. 671–689, 2019.

[28] M. Narendran and P. Prakasam, "An energy aware competition based clustering for cluster head selection in wireless sensor network with mobility," *Cluster Computing*, vol. 22, no. 5, pp. 11 019–11 028, 2019.

[29] S. Vural, P. Navaratnam, N. Wang, and R. Tafazolli, "Asynchronous clustering of multihop wireless sensor networks," in *2014 IEEE International Conference on Communications (ICC)*, 2014, pp. 472–477.

[30] M. Bateni, A. Bhaskara, S. Lattanzi, and V. S. Mirrokni, "Distributed balanced clustering via mapping coresets." in *NIPS*, 2014, pp. 2591–2599.

[31] O. Bachem, M. Lucic, and A. Krause, "Scalable k-means clustering via lightweight coresets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1119–1127.

[32] A. Soliman, S. Girdzijauskas, M.-R. Bouguelia, S. Pashami, and S. Nowaczyk, "Decentralized and adaptive k-means clustering for non-iid data using hyperloglog counters," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2020, pp. 343–355.

[33] S. Heule, M. Nunkesser, and A. Hall, "Hyperloglog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm," in *Proceedings of the 16th International Conference on Extending Database Technology*, 2013, pp. 683–692.

[34] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and

I. Perona, "An extensive comparative study of cluster validity indices," *Pattern Recognition*, vol. 46, no. 1, pp. 243–256, 2013.

[35] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.

[36] J. A. Bilmes *et al.*, "A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," *International Computer Science Institute*, vol. 4, no. 510, p. 126, 1998.

[37] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[38] O. Bachem, M. Lucic, S. H. Hassani, and A. Krause, "Approximate k-means++ in sublinear time," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[39] D. Arthur and S. Vassilvitskii, "How slow is the k-means method?" in *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*. Association for Computing Machinery, 2006, p. 144–153.

[40] A. Sharma and K. K. Paliwal, "Fast principal component analysis using fixed-point algorithm," *Pattern Recognition Letters*, vol. 28, no. 10, pp. 1151–1155, 2007.

[41] I. M. Johnstone and A. Y. Lu, "Sparse principal components analysis," *arXiv preprint arXiv:0901.4392*, 2009.

[42] R. C. Pinto and P. M. Engel, "A fast incremental gaussian mixture model," *PloS one*, vol. 10, no. 10, p. e0139931, 2015.

[43] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian processes for big data," in *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'13. AUAI Press, 2013, p. 282–290.

[44] R. Zivan and A. Meisels, "Message delay and discsp search algorithms," *Annals of Mathematics and Artificial Intelligence*, vol. 46, no. 4, pp. 415–439, 2006.

[45] H. Cherifi, G. Palla, B. K. Szymanski, and X. Lu, "On community structure in complex networks: challenges and opportunities," *Applied Network Science*, vol. 4, no. 1, pp. 1–35, 2019.

[46] L. Fu and E. Medico, "Flame, a novel fuzzy clustering method for the analysis of DNA microarray data," *BMC Bioinform.*, vol. 8, 2007.

[47] https://CRAN.R-project.org/package=mlbench.

[48] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, pp. 1–30, 2007.

Cheng Qiao Cheng Qiao is a Postdoc reseacher in Guangzhou University. He received the Ph.D degree from University College Cork, Ireland, in 2021. He completed his M.S. from Fujian Normal University, China in 2013 and B.E from Xidian University, China in 2010. Before joined UCC, he worked as Research Assistant in Shenzhen Institute of Advanced Technology, Chinese Academy of Science (CAS). His research interests include clustering algorithms, distributed autonomous networks, and distributed learning in Wireless Networks.

Kenneth N. Brown Kenneth N. Brown is a Professor of Computer Science at University College Cork in Ireland. He joined UCC Computer Science Department as a senior lecturer in 2003. Prior to that he was a lecturer at the University of Aberdeen, a Research Fellow at Carnegie Mellon University, and a Research Associate at the University of Bristol. His research interests are in the application of AI, optimisation and distributed reasoning, with a particular focus on wireless networks. He is a Co-Principal Investigator/Group Leader in Insight: Centre for Data Analytics.

Fan Zhang Fan Zhang is currently an associate professor in Guangzhou University. He was a research associate in the School of Computer Science and Engineering, University of New South Wales. He received the BEng degree from Zhejiang University in 2014, and the PhD from University of Technology Sydney in 2017. His research interests include graph algorithms and social networks. Since 2017, he has published more than 20 papers in top venues including SIGMOD, PVLDB, ICDE, IJCAI, AAAI, TKDE and VLDB Journal.

Zhihong Tian Zhihong Tian is currently a Professor, and Dean, with the Cyberspace Institute of Advanced Technology, Guangzhou University, Guangdong Province, China. Guangdong Province Universities and Colleges Pearl River Scholar (Distinguished Professor). He is also a part-time Professor at Carlton University, Ottawa, Canada. Previously, he served in different academic and administrative positions at the Harbin Institute of Technology. He has authored over 200 journal and conference papers in these areas. His research interests include computer networks and cyberspace security. His research has been supported in part by the National Natural Science Foundation of China, National Key Research and Development Plan of China, National High tech RD Program of China (863 Program), and National Basic Research Program of China (973 Program). He also served as a member, Chair, and General Chair of a number of international conferences. He is a Senior Member of the China Computer Federation, and a Senior Member of IEEE.